

# C파일 빌드 과정, 정렬 알고리즘, 디버깅

---

지능기전공학부 스마트기기공학전공 3학년 정건희  
2023-1

# 1-1. 전체적인 빌드 과정



## 1-2. 컴파일 과정

### .c파일 = 소스파일

```
#pragma warning(disable:4996)
#include<stdio.h>

int main() {
    printf("Hello World");
    return 0;
}
```

 Hello world를 출력해라

소스파일은 사람이 이해할 수 있는 소스코드가 들어있는 파일

## 1-2. 컴파일(compile) 과정

**.obj파일 = .o 파일 = 오브젝트파일**

```
00000000100401080 <main>:
100401080: 55                push    %rbp
100401081: 48 89 e5          mov     %rsp,%rbp
100401084: 48 83 ec 20       sub     $0x20,%rsp
100401088: e8 33 00 00 00    call   1004010c0 <__main>
10040108d: 48 8d 05 6c 1f 00 00 lea     0x1f6c(%rip),%rax    # 100403000 <.rdata>
100401094: 48 89 c1          mov     %rax,%rcx
100401097: e8 34 00 00 00    call   1004010d0 <printf>
10040109c: b8 00 00 00 00    mov     $0x0,%eax
1004010a1: 48 83 c4 20       add     $0x20,%rsp
1004010a5: 5d                pop     %rbp
1004010a6: c3                ret
1004010a7: 90                nop
1004010a8: 90                nop
1004010a9: 90                nop
1004010aa: 90                nop
1004010ab: 90                nop
1004010ac: 90                nop
1004010ad: 90                nop
1004010ae: 90                nop
1004010af: 90                nop
```

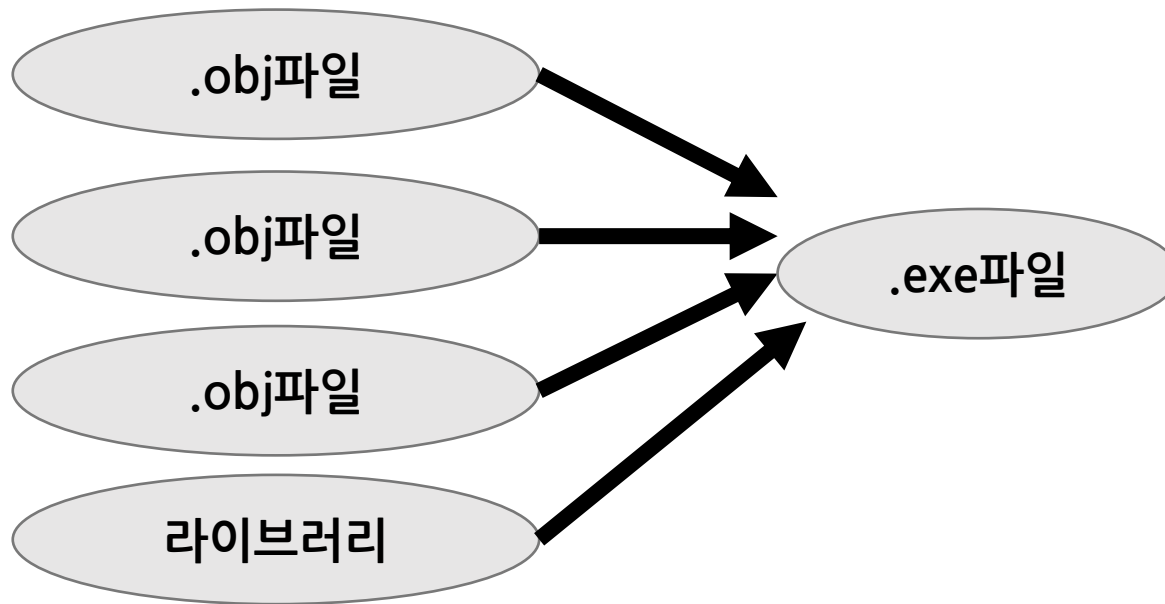
**소스파일을 기계가 알아 들을 수 있는 기계어로 번역한 파일**



그럼 컴파일러(Compiler)는 무엇일까?

## 1-3. 링킹(linking) 과정

**obj파일들과 라이브러리들을 결합하여**  
**최종적인 실행 파일 1개를 생성하는 과정**



## 1-3. 링킹(linking) 과정-라이브러리

자주 쓰는 함수 등을 미리 구현해 놓은 것 (Library)  
사용할 라이브러리의 헤더파일을 include 해서 사용

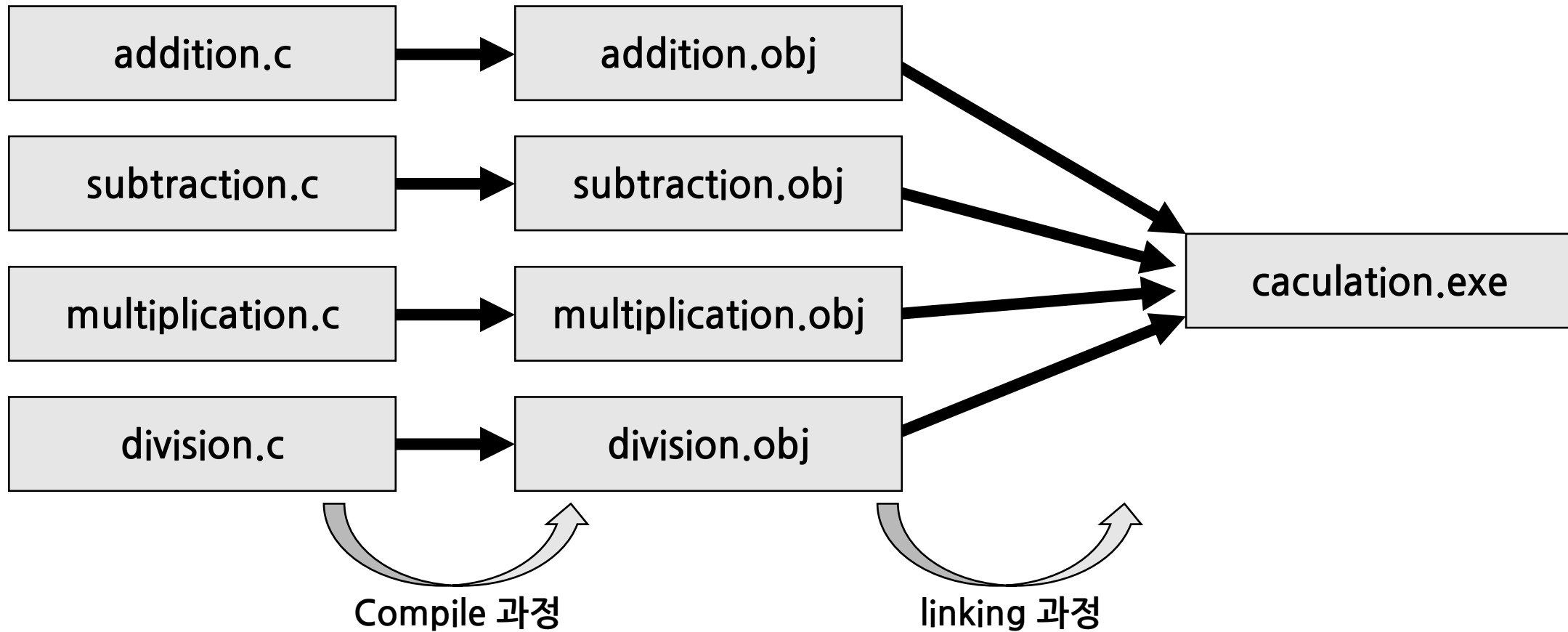
```
#include<stdio.h>
```

printf();

scanf();

cf) 고급c에서 <string.h>와 <stdlib.h> 배울 예정

## 1-4. 빌드(build) 과정



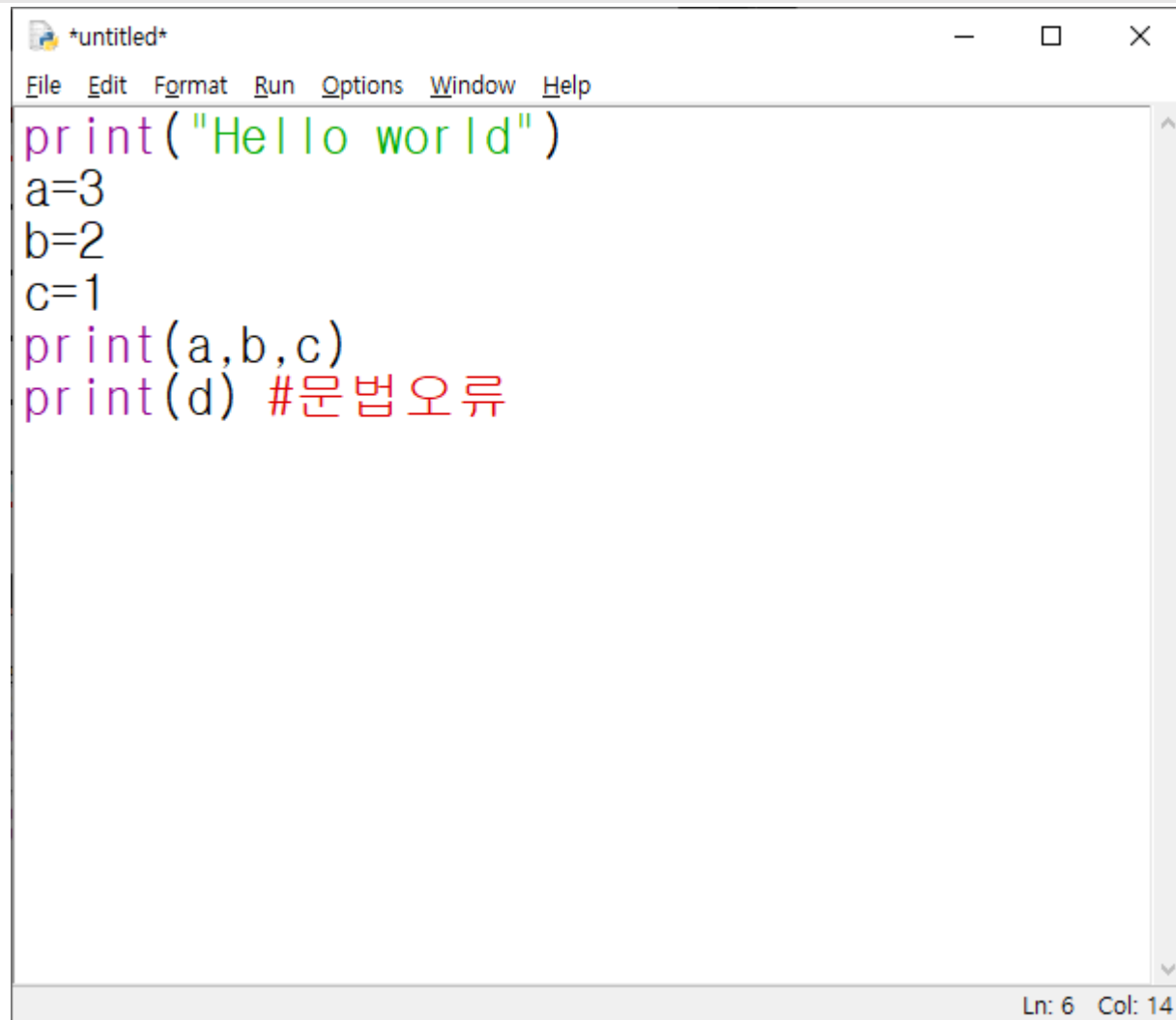
# 컴파일 언어

## 1-5. 인터프리터 언어

```
#pragma warning(disable:4996)
#include<stdio.h>

int main() {

    printf("Hello World");
    int a = 3;
    int b = 2;
    int c = 1;
    printf("%d %d %d", a, b, c);
    printf("%d", d);
    return 0;
}
```



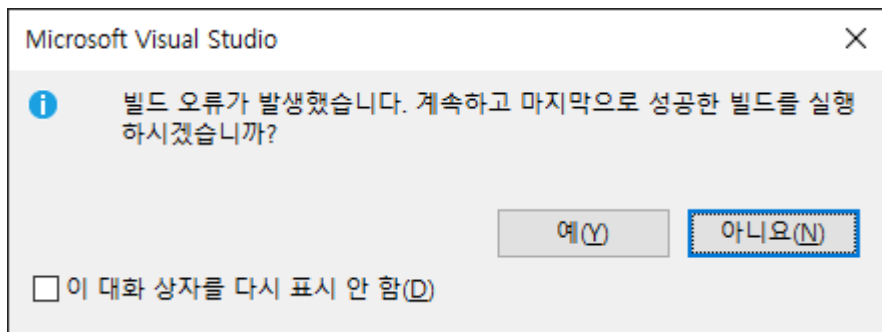
```
*untitled*
File Edit Format Run Options Window Help

print("Hello world")
a=3
b=2
c=1
print(a,b,c)
print(d) #문법오류

Ln: 6 Col: 14
```



## 1-5. 인터프리터 언어



빌드가 안되어서 exe파일이 만들어지지 못함

```
IDLE Shell 3.11.3
File Edit Shell Debug Options Window Help
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:
59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
ore information.
>>>
===== RESTART: C:/Users/MisterJerry/Desktop/coding
mentoring.py =====
Hello world
3 2 1
Traceback (most recent call last):
  File "C:/Users/MisterJerry/Desktop/coding_mentoring.py", line 6, in <module>
    print(d) #문법오류
NameError: name 'd' is not defined. Did you mean: 'id'?
>>>
```

문법 오류인 부분 전까지는 실행됨

## 1-5. 인터프리터 언어

python 처럼 **1줄 1줄 실행**되는 언어가 인터프리터 언어

## 2-1. 정렬 알고리즘 종류

버블정렬 (Bubble sort)

삽입정렬 (Insertion sort)

선택정렬 (selection sort)

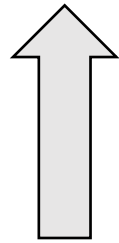
병합정렬 (merge sort)

퀵정렬 (Quick sort)

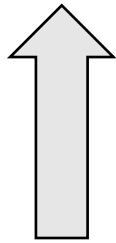
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
700	600	500	400	300	200	100



idx1



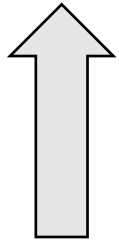
idx2

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

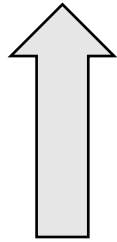
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
600	700	500	400	300	200	100



idx1



idx2

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

## 2-2. 버블 정렬

```
int arr[7];
```

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
500	700	600	400	300	200	100

ixd 1

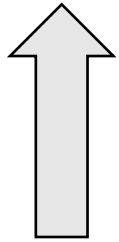
idx2

```
if (arr[idx1] > arr[idx2]) {
    int tmp = arr[idx1];
    arr[idx1] = arr[idx2];
    arr[idx2] = tmp;
}
```

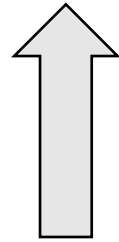
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
400	700	600	500	300	200	100



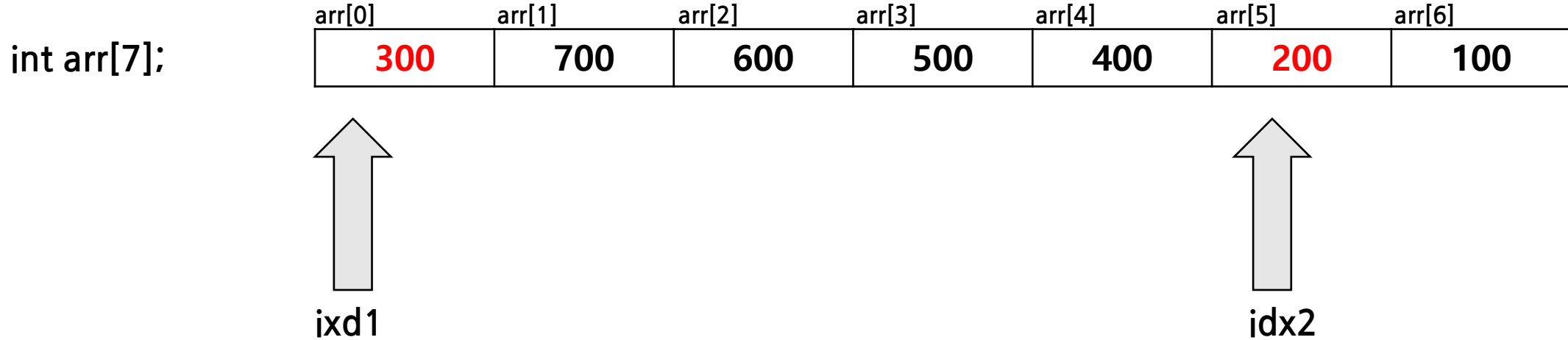
idx1



idx2

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

## 2-2. 버블 정렬



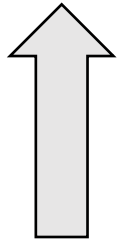
```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```



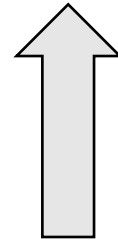
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
200	700	600	500	400	300	100



idx1



idx2

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

## 2-2. 버블 정렬

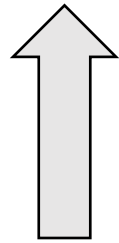
	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
int arr[7];	100	700	600	500	400	300	200

제일 작은 값이 위치하게 됨

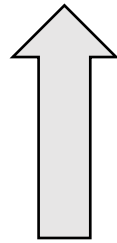
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
100	700	600	500	400	300	200



idx1



idx2

... 반복

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

## 2-2. 버블 정렬

int arr[7];

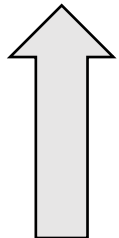
arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
100	200	700	600	500	400	300

```
if (arr[idx1] > arr[idx2]) {  
    int tmp = arr[idx1];  
    arr[idx1] = arr[idx2];  
    arr[idx2] = tmp;  
}
```

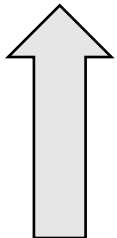
## 2-2. 버블 정렬

int arr[7];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]
700	600	500	400	300	200	100



i



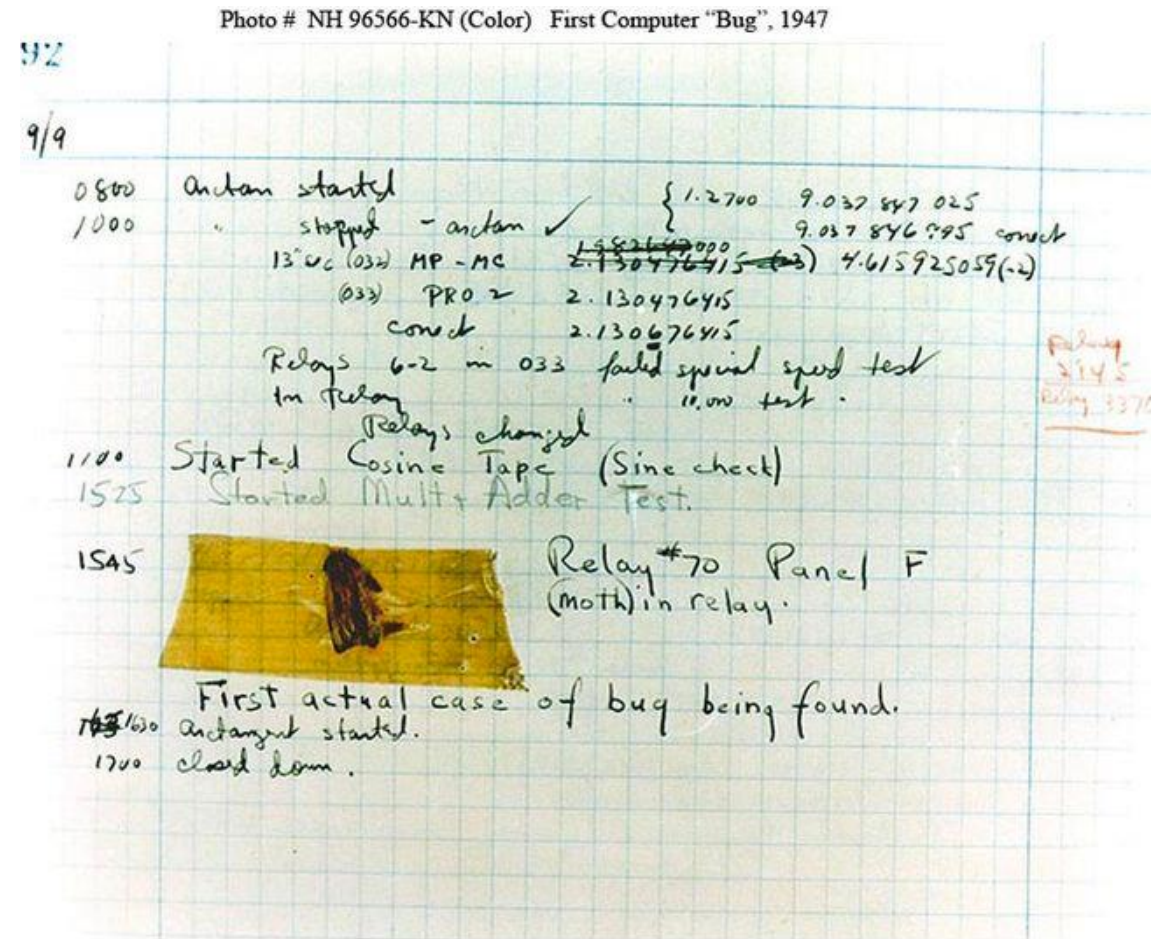
j

```
for (int i = 0; i < 7; i++) {  
    for (int j = i + 1; j < 7; j++) {  
        if (arr[i] > arr[j]) {  
            int tmp = arr[i];  
            arr[i] = arr[j];  
            arr[j] = tmp;  
        }  
    }  
}
```

## 3-1. 디버깅(Debugging)이란?



Grace Hopper(1906~1992)



마크 II 컴퓨터의 메인 프레임에 오류원인이 나방의 날개로 밝혀져

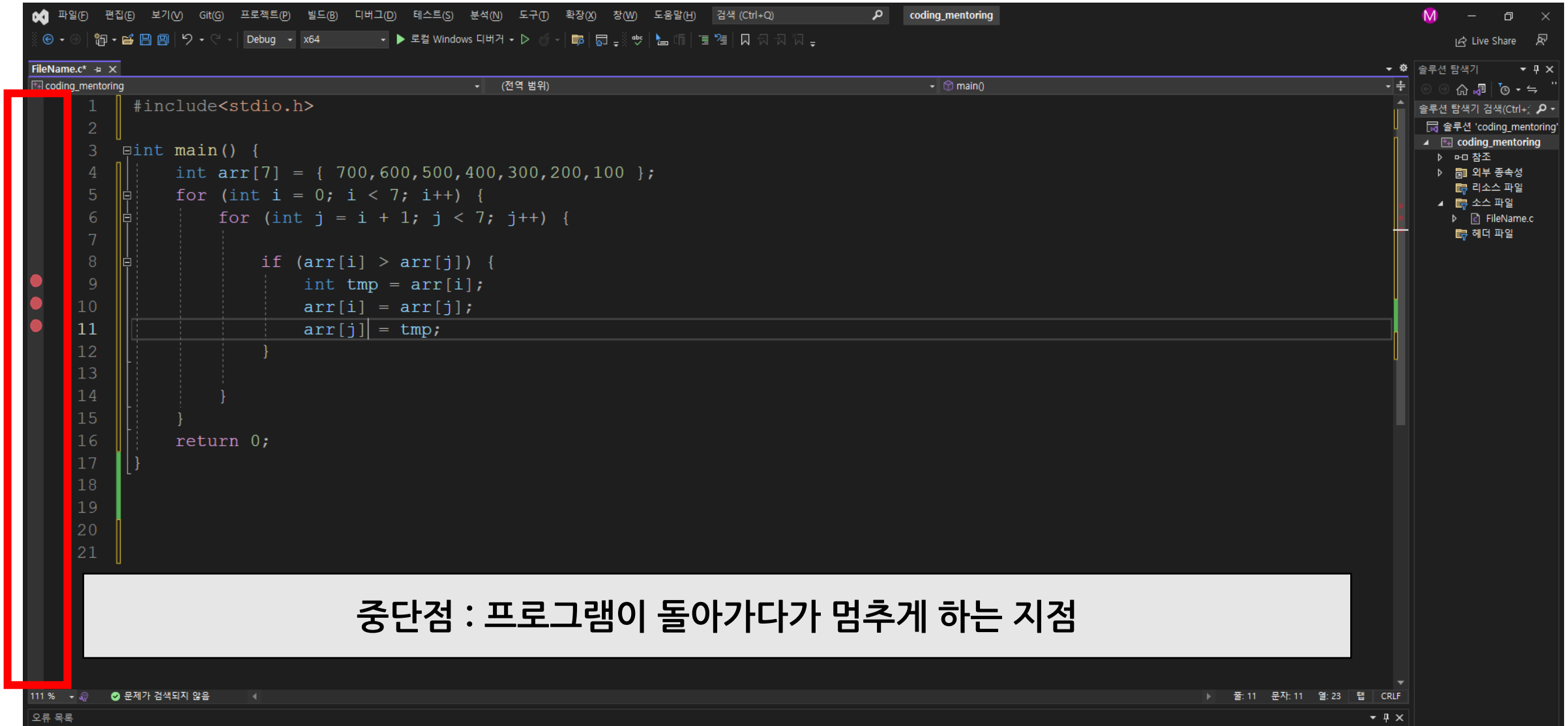
## 3-1. 디버깅(Debugging)이란?

이 사건을 계기로 버그는 프로그램의 결함 또는 오류를 칭하는 말이 됨

디버그란?

컴퓨터 프로그램에서 발생하는 오류를 찾는 것

## 3-2. Visual Studio 디버깅 - 중단점



The screenshot shows the Visual Studio IDE with a C program open. The code is as follows:

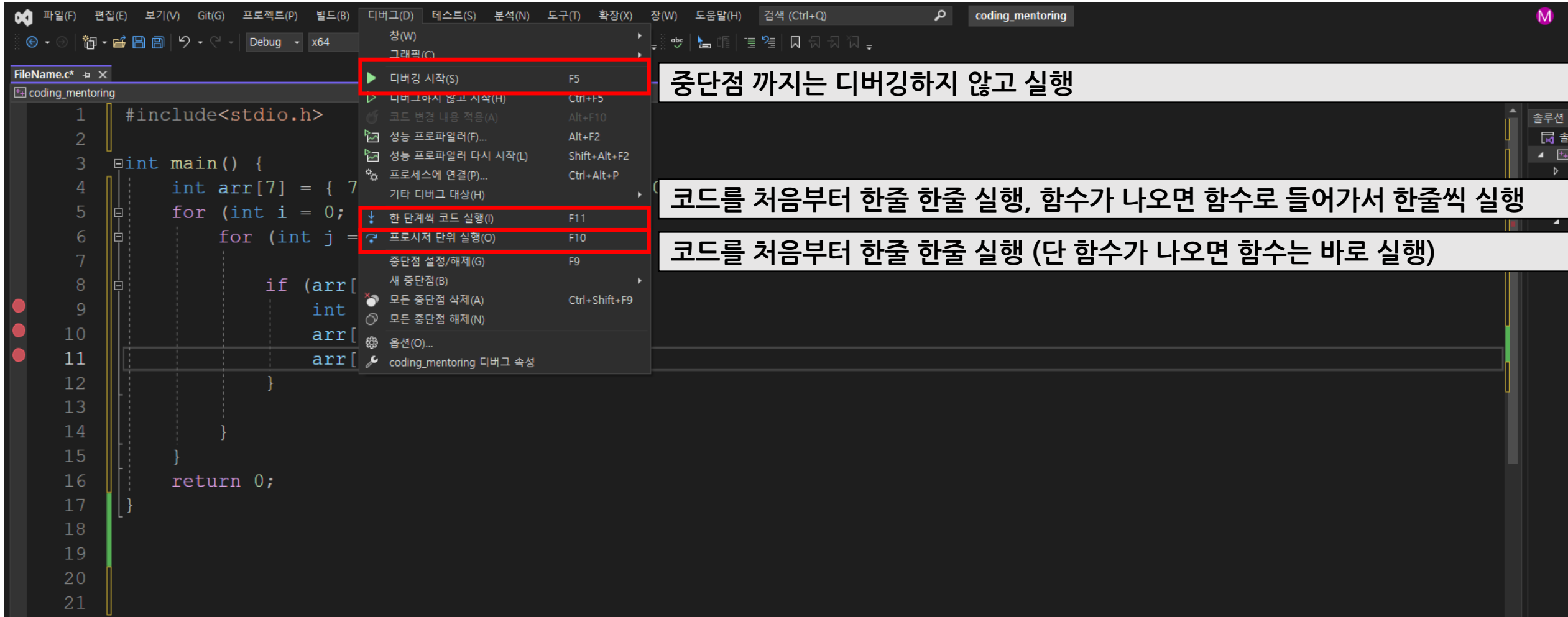
```
1  #include<stdio.h>
2
3  int main() {
4      int arr[7] = { 700,600,500,400,300,200,100 };
5      for (int i = 0; i < 7; i++) {
6          for (int j = i + 1; j < 7; j++) {
7
8              if (arr[i] > arr[j]) {
9                  int tmp = arr[i];
10                 arr[i] = arr[j];
11                 arr[j] = tmp;
12             }
13         }
14     }
15     return 0;
16 }
17
18
19
20
21
```

A red box highlights the left margin of the code editor, where a breakpoint (red dot) is set on line 11. The right sidebar shows the 'Solution Explorer' with the project 'coding\_mentoring' and its files: '외부 종속성', '리소스 파일', '소스 파일' (containing 'FileName.c'), and '헤더 파일'.

중단점 : 프로그램이 돌아가다가 멈추게 하는 지점



## 3-2. Visual Studio 디버깅 옵션



The screenshot shows the Visual Studio interface with the 'Debug' menu open. The menu options are as follows:

- 창(W)
- 그래픽(G)
- ▶ 디버깅 시작(S) F5
- ▶ 디버깅하지 않고 시작(H) Ctrl+F5
- ▶ 코드 변경 내용 적용(A) Alt+F10
- ▶ 성능 프로파일러(F)...
- ▶ 성능 프로파일러 다시 시작(L) Shift+Alt+F2
- ▶ 프로세스에 연결(P)...
- ▶ Ctrl+Alt+P
- ▶ 기타 디버그 대상(H)
- ▶ 한 단계씩 코드 실행(I) F11
- ▶ 프로시저 단위 실행(O) F10
- ▶ 중단점 설정/해제(G) F9
- ▶ 새 중단점(B)
- ▶ 모든 중단점 삭제(A) Ctrl+Shift+F9
- ▶ 모든 중단점 해제(N)
- ▶ 옵션(O)...
- ▶ coding\_mentoring 디버그 속성

Three red boxes highlight the following options:

- ▶ 디버깅 시작(S) F5
- ▶ 한 단계씩 코드 실행(I) F11
- ▶ 프로시저 단위 실행(O) F10

Three text boxes provide instructions for each highlighted option:

- 중단점까지는 디버깅하지 않고 실행** (Run without debugging until the breakpoint)
- 코드를 처음부터 한줄 한줄 실행, 함수가 나오면 함수로 들어가서 한줄씩 실행** (Run code line by line from the beginning, step into functions)
- 코드를 처음부터 한줄 한줄 실행 (단 함수가 나오면 함수는 바로 실행)** (Run code line by line from the beginning, but skip into functions)

## 3-2. Visual Studio 디버깅 - 조사식

The screenshot illustrates the Visual Studio IDE during a debugging session. The main editor shows a C program with a `print()` function and a `main()` function. The `print()` function is currently selected, and the 'Debug' menu is open, showing the 'Watch' option. The 'Watch' window displays the 'print()' function call, and the 'Call Stack' window shows the sequence of function calls leading to the current state.

**File Name:** coding\_mentoring.c

```

1 #include<stdio.h>
2 void print() {
3     printf("Hello");
4 }
5 int main() {
6     int arr[7] = { 7
7     for (int i = 0;
8     for (int j =
9
10    if (arr[
11        int
12        arr[
13        arr[
14    }
15
16    }
17
18
19    print();
20    return 0;
21 }
22
23
24

```

**Debug Menu:**

- 디버그(D)
  - 장(W)
    - 중단점(B) Ctrl+Alt+B
    - 예외 설정(X) Ctrl+Alt+E
    - 출력(O)
    - 진단 도구 표시(T) Ctrl+Alt+F2
    - GPU 스레드(U)
    - 작업(S) Ctrl+Shift+D, K
    - 병렬 스택(K) Ctrl+Shift+D, S
    - 병렬 조사식(R)
      - 조사식(W)
        - 조사식 1(1) Ctrl+Alt+W, 1
        - 조사식 2(2) Ctrl+Alt+W, 2
        - 조사식 3(3) Ctrl+Alt+W, 3
        - 조사식 4(4) Ctrl+Alt+W, 4

**Watch Window:**

이름	값	형식
출력	출력	출력

**Call Stack Window:**

이름	주소	프레임
출력	출력	출력


## 3-3. 버블 정렬 코드를 디버깅하면서 실습

```
#include<stdio.h>

int main() {
    int arr[7] = { 700, 600, 500, 400, 300, 200, 100 };
    for (int i = 0; i < 7; i++) {
        for (int j = i + 1; j < 7; j++) {
            if (arr[i] > arr[j]) {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
    }

    return 0;
}
```

- F10으로 코드를 하나하나 실행
- 조사식에는 arr와 arr[i], arr[j]를 넣어서 arr배열이 바뀌는 과정 확인

조사식 1	
검색(Ctrl+E)  < > 검색 심도: 3	
이름	값
▸ arr	0x0000005ef10ffa98 {700, 600, 500, 400, 300, 200, 100}
arr[i]	700
arr[j]	600
감시할 항목 추가	
자동 로컬 조사식 1	

## 4. QnA



Q&A